Cache Contention and Application Performance Prediction for Multi-Core Systems

Chi Xu*, Xi Chen, Robert P. Dick, Zhuoqing Morley Mao

*University of Minnesota, University of Michigan

IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS), March 2010

Motivation

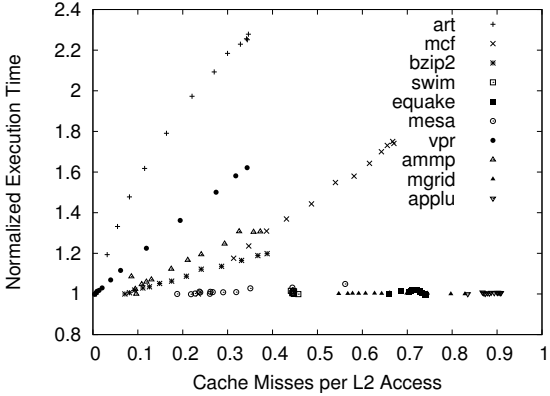
Multiprocessor architectures (CMP)
with shared last-level caches

- \+ Inter-process communication
- \+ Heterogeneous cache allocation
- − Contention

Motivation

Multiprocessor architectures (CMP) with shared last-level caches

- + Inter-process communication
- + Heterogeneous cache allocation
- − Contention



⇒ Performance implications of core assignment

Goal

- Model cache contention
- Easy and automatic
- No modifications to existing hardware or operating system
- No exhaustive offline simulation
- Complementary to existing work

Analytical Model — System

- N-core processor
- On-chip last-level L2 Cache
  - Set-associative (*ways* = *lines* per *set*)
  - LRU replacement policy
  - Shared among cores
  - No Prefetching
- Applications in steady state

Analytical Model — Applications I

Effective Cache Size

$$\sum_{i=1}^{N} S_i = A$$

N Total number of processes

S_i Effective cache size of process i (ways occupied by i)

A Associativity of cache

Analytical Model — Applications I

Effective Cache Size

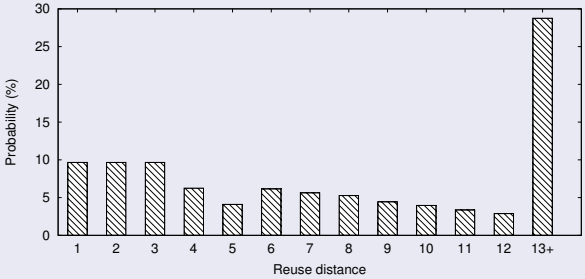
$$\sum_{i=1}^{N} S_i = A$$

N Total number of processes

S_i Effective cache size of process i (ways occupied by i)

A Associativity of cache

Reuse Distance



Analytical Model — Applications I

Effective Cache Size

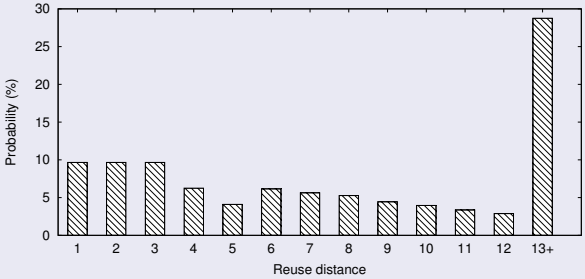
$$\sum_{i=1}^{N} S_i = A$$

N Total number of processes

S_i Effective cache size of process i (ways occupied by i)

A Associativity of cache

Reuse Distance



$$\mathrm{MPA}_i(S_i) = \int_{S_i}^{\infty} \mathrm{hist}_i(x)\,\mathrm{d}x$$

MPA Probability of cache miss for process i

hist Linear interpolation of reuse distance histgram

Analytical Model — Applications II

Cache Accesses

$$\mathrm{APS} = \frac{\mathrm{API}}{\mathrm{SPI}}$$

APS Accesses per second

API Accesses per instruction (fixed for each application)

SPI Seconds per instruction

Analytical Model — Applications II

Cache Accesses

$$\mathrm{APS} = \frac{\mathrm{API}}{\mathrm{SPI}}$$

- APS Accesses per second
- API Accesses per instruction (fixed for each application)
- SPI Seconds per instruction

$$\mathrm{SPI} = \alpha \cdot \mathrm{MPA} + \beta$$

- α Off-chip latency (memory, disk)
- β On-chip latency (computation)

Analytical Model — Applications II

Cache Accesses

$$\mathrm{APS} = \frac{\mathrm{API}}{\mathrm{SPI}}$$

- APS Accesses per second
- API Accesses per instruction (fixed for each application)
- SPI Seconds per instruction

$$\mathrm{SPI} = \alpha \cdot \mathrm{MPA} + \beta$$

- α Off-chip latency (memory, disk)
- β On-chip latency (computation)

Effective Cache Size

$$G_i(n) = \sum_{s=1}^{n} (P_{s,n} \cdot s)$$

- $G_i(n)$ Effective cache size of process i after n accesses
- $P_{s,n}$ Probability of having s cache lines after n consecutive accesses

Analytical Model — Applications II

Cache Accesses

$$\mathrm{APS} = \frac{\mathrm{API}}{\mathrm{SPI}}$$

- APS Accesses per second
- API Accesses per instruction (fixed for each application)
- SPI Seconds per instruction

$$\mathrm{SPI} = \alpha \cdot \mathrm{MPA} + \beta$$

- α Off-chip latency (memory, disk)
- β On-chip latency (computation)

Effective Cache Size

$$G_i(n) = \sum_{s=1}^{n} (P_{s,n} \cdot s)$$

steady $\Downarrow$ state

$$n = G_i^{-1}(S_i)$$

- $G_i(n)$ Effective cache size of process i after n accesses
- $P_{s,n}$ Probability of having s cache lines after n consecutive accesses

Analytical Model — Applications III

At time t there is a duration T such that data accessed...

- before $t - T$ are evicted from cache
- during $[t - T, t]$ are present in cache

Analytical Model — Applications III

At time t there is a duration T such that data accessed...

- before $t - T$ are evicted from cache
- during $[t - T, t]$ are present in cache

Assuming all processes are in steady state:

$$S_i = G_i(\mathrm{APS}_i \cdot T)$$

Analytical Model — Applications III

At time t there is a duration T such that data accessed...

- before $t - T$ are evicted from cache
- during $[t - T, t]$ are present in cache

Assuming all processes are in steady state:

$$S_i = G_i(\mathrm{APS}_i \cdot T)$$

$$\Downarrow$$

$$\mathrm{APS}_i = G_i^{-1}(S_i)/T$$

Analytical Model — Applications III

At time t there is a duration T such that data accessed...

- before $t - T$ are evicted from cache
- during $[t - T, t]$ are present in cache

Assuming all processes are in steady state:

$$S_i = G_i(\mathrm{APS}_i \cdot T)$$

$$\Downarrow$$

$$\mathrm{APS}_i = G_i^{-1}(S_i)/T$$

$$\Downarrow$$

$$\mathrm{APS}_i = \frac{G_i^{-1}(S_i)}{T} = \frac{\mathrm{API}_i}{\alpha_i \cdot \mathrm{MPA}_i(S_i) + \beta_i}$$

Reminder

$$\mathrm{APS} = \frac{\mathrm{API}}{\mathrm{SPI}}$$

$$\mathrm{SPI} = \alpha \cdot \mathrm{MPA} + \beta$$

Analytical Model — Applications III

At time t there is a duration T such that data accessed...

- before $t - T$ are evicted from cache
- during $[t - T, t]$ are present in cache

Assuming all processes are in steady state:

$$S_i = G_i(\mathrm{APS}_i \cdot T)$$

$$\Downarrow$$

$$\mathrm{APS}_i = G_i^{-1}(S_i)/T$$

$$\Downarrow$$

$$\mathrm{APS}_i = \frac{G_i^{-1}(S_i)}{T} = \frac{\mathrm{API}_i}{\alpha_i \cdot \mathrm{MPA}_i(S_i) + \beta_i}$$

$$\Downarrow$$

$$T = \frac{G_i^{-1}(S_i) \cdot \alpha_i \cdot \mathrm{MPA}_i(S_i) + \beta_i}{\mathrm{API}_i}$$

Reminder

$$\mathrm{APS} = \frac{\mathrm{API}}{\mathrm{SPI}}$$

$$\mathrm{SPI} = \alpha \cdot \mathrm{MPA} + \beta$$

Analytical Model — Applications III

At time t there is a duration T such that data accessed...

- before $t - T$ are evicted from cache
- during $[t - T, t]$ are present in cache

Assuming all processes are in steady state:

$$S_i = G_i(\mathrm{APS}_i \cdot T)$$

$$\Downarrow$$

$$\mathrm{APS}_i = G_i^{-1}(S_i)/T$$

$$\Downarrow$$

$$\mathrm{APS}_i = \frac{G_i^{-1}(S_i)}{T} = \frac{\mathrm{API}_i}{\alpha_i \cdot \mathrm{MPA}_i(S_i) + \beta_i}$$

$$\Downarrow$$

$$T = \frac{G_i^{-1}(S_i) \cdot \alpha_i \cdot \mathrm{MPA}_i(S_i) + \beta_i}{\mathrm{API}_i}$$

Reminder

$$\mathrm{APS} = \frac{\mathrm{API}}{\mathrm{SPI}}$$

$$\mathrm{SPI} = \alpha \cdot \mathrm{MPA} + \beta$$

$$\sum_{i=1}^{N} S_i = A$$

$$\Downarrow$$

$$\forall_{j=1}^{N} : \frac{G_1^{-1}(S_1)}{G_j^{-1}(S_j)} - \frac{\mathrm{API}_1 \cdot (\alpha_j \cdot \mathrm{MPA}_j(S_j) + \beta_j)}{\mathrm{API}_i \cdot (\alpha_1 \cdot \mathrm{MPA}_1(S_1) + \beta_1)} \text{ and } \sum_{i=1}^{N} S_i - A = 0$$

Automated Profiling

- Two processes running on separate cores sharing A-way last-level cache
- One process uses l ways $\Rightarrow$ other process uses $A - l$ ways
- *stressmark*: synthetic application with configurable cache occupation
- Gather information on API, MPA and SPI via hardware performance counters
- Derive reuse distance histogram, effective cache size (S), α and β

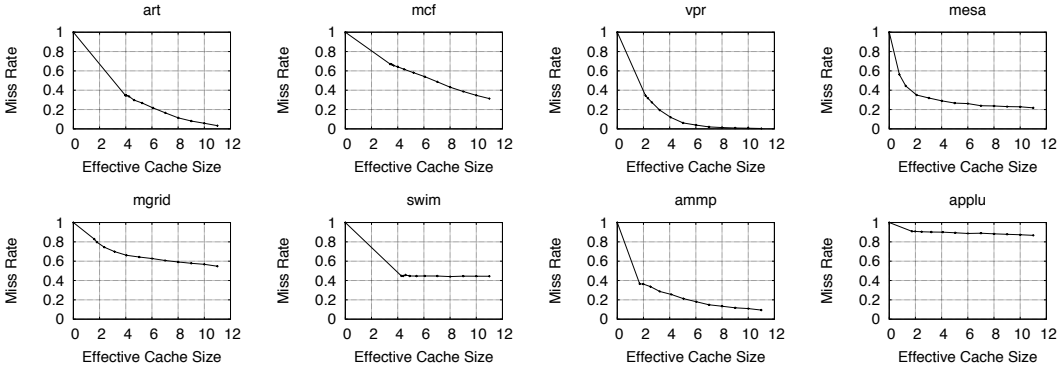
$\Rightarrow$ application-dependent feature vector

Evaluation

- Intel Core 2 Duo-P8600 (2 core @ 2.4GHz, 3 MB 12-way associative L2 cache)
- MacOS X 10.5
- Profiling via *Shark* at a period of 2 ms
- Subset of SPEC CPU2000: 5 CPU-intensive + 5 memory-intensive
- Each application run 12 times for 10 s to determine characteristics
- Examine all 55 pairwise combinations

Application profiles

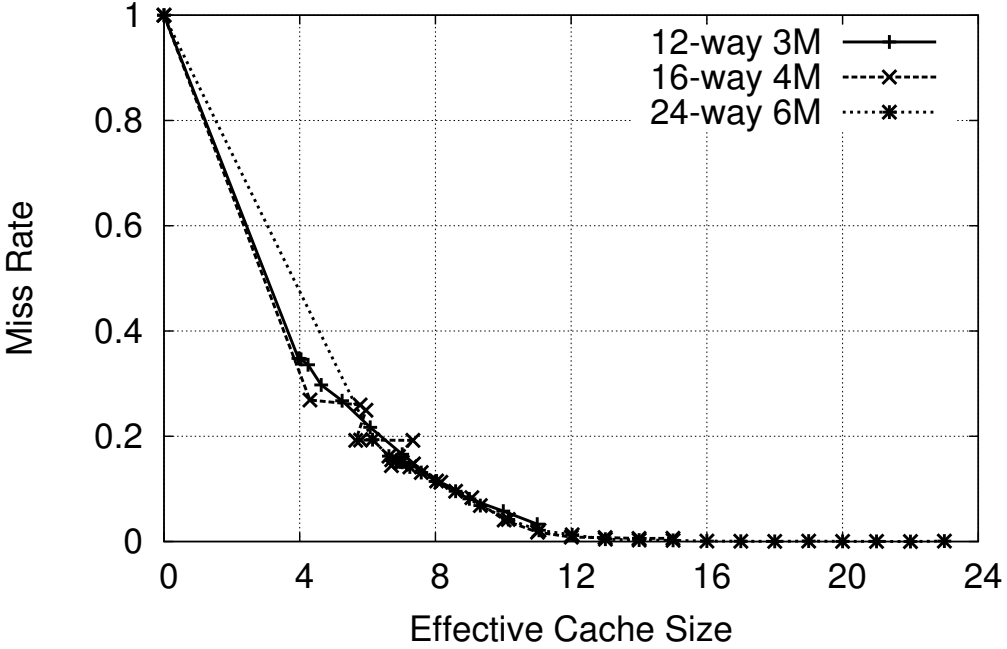
Benchmark	art	mcf	bzip2	swim	equake	mesa	vpr	ammp	mgrid	applu
API	0.0225	0.0733	0.0044	0.0116	0.0074	0.0013	0.0102	0.0092	0.0018	0.0018
α ($\times 10^{-9}$)	446	134	99.9	-99.6	60.5	30.7	306	243	0.609	3.12
β ($\times 10^{-7}$)	1.34	5.86	1.50	1.97	2.28	1.55	1.65	1.83	1.28	1.15



Prediction Accuracy

Benchmark	CAMP				AB				MB			
	MPA		SPI		MPA		SPI		MPA		SPI	
	Error (%)	>5% (%)	Error (%)	>5% (%)	Error (%)	>5% (%)	Error (%)	>5% (%)	Error (%)	>5% (%)	Error (%)	>5% (%)
art	1.61	0	3.68	40	4.60	50	10.26	80	5.88	70	18.09	90
vpr	0.88	0	1.48	0	4.70	40	7.67	60	5.89	30	9.24	50
mcf	2.10	10	3.70	20	2.82	10	3.97	40	6.79	40	7.72	70
ammp	2.82	20	3.04	20	4.03	30	4.16	30	5.89	60	6.78	90
bzip2	1.86	10	1.17	0	3.17	20	1.89	0	6.09	60	3.63	30
mesa	4.23	50	0.83	0	4.90	30	0.94	0	7.77	50	1.55	0
swim	0.28	0	0.86	0	0.23	0	0.81	0	0.27	0	0.78	0
equake	0.70	0	0.38	0	0.92	0	0.41	0	1.43	0	0.45	0
applu	1.13	0	0.32	0	0.86	0	0.31	0	1.79	10	0.33	0
mgrid	2.79	10	0.28	0	3.35	20	0.28	0	6.00	40	0.30	0
top 5 average	1.86	8	2.61	16	3.86	30	5.59	42	6.11	52	9.09	66
average	1.86	4	1.57	8	2.94	20	3.07	21	4.78	36	4.89	33

Generality — *art*



Conclusion

Summary

- Predictive model of contention on shared last-level cache
- Automated profiling and extraction of feature vector
- No modification of hardware or operating system
- "Average" error of $<1.6\%$

Conclusion

Summary

- Predictive model of contention on shared last-level cache
- Automated profiling and extraction of feature vector
- No modification of hardware or operating system
- "Average" error of $<1.6\%$

Discussion

- Varying input data
- Benchmarking crimes
- Generalisation
- Practical application